1997

A Taxonomy for Networked Virtual Environments

Macedonia, Michael

þÿ M a c e d o n i a , M i c h a e l a n d Z y d a , M i c h a e l A T a x o n o m y f o r N e t w o r k e d V i r t u a
IEEE Multimedia, Volume 4, No. 1, January - March 1997, pp. 48-56.
http://hdl.handle.net/10945/41565

A Taxonomy for Networked Virtual Environments

Michael R. Macedonia
Fraunhofer Center for Research in Computer Graphics
167 Angell St.
Providence, R.I. 02906
+1-401-453-6363
mmacedon@crcg.edu

Michael J. Zyda
Computer Science Department
Naval Postgraduate School
Monterey, California 93943-5118 USA
+1-408-656-2305
zyda@siggraph.org

ABSTRACT

The development of multi-user networked virtual worlds has become a major area of interest in the computer and communications fields. However, there has been little effort to provide a framework for understanding distributed virtual environments (VEs). In this paper we discuss VEs in the context of how network communications, views, data, and processes are distributed while emphasizing those aspects critical to scaling environments. We find most of the systems described in this paper scale to accommodate a handful of users. We also discuss why systems which demand strong data consistency, causality, and reliable communications at the same time while supporting real-time interaction are not likely to scale very well. Furthermore, if the systems are to be geographically dispersed, then high-speed, multicast communication is required.
.

KEYWORDS: Virtual Reality, Distributed Interactive Simulation, Internet Protocol Multicast, Distributed Interactive Entertainment, Large-scale Virtual Environments.

OVERVIEW

In this paper we discuss what needs to be considered when building large-scale VEs. This frame of reference is necessary to have a common understanding of the many components of distributed VE systems.

Currently, there are relatively few examples of academic VE systems in which to apply the framework. Research into large-scale distributed virtual worlds has been limited be-

cause of a number of practical factors. Immature network technology has relegated most distributed VEs to Ethernet local area networks (LAN). Large-scale VEs will need to use wide-area-networks (WANs) to expand both their physical geographic scope and number of participating hosts. Furthermore, until recently, real-time graphics performance had been confined to specialized and very expensive computer image generators. Software development and graphics databases have also progressed slowly and the interfaces for immersing the human into the environment have been primitive at best [9].

These problems are being overcome to a limited degree by the rapid growth of high-speed internetworks, the availability of low-cost, off-the-shelf graphics workstations, and the development of standard graphics tools and libraries. Moreover, distributed VE's hold the promise for new educational, training, entertainment, and medical applications. For instance, over twenty companies, including Microsoft, are developing 3D computer gaming networks.

However, there are many challenges yet to be met that would appear trivial if these applications were not distributed across wide-area networks.

TAXONOMY

Virtual environments mimic many aspects of operating systems. For example, the now defunct system developed by the Human Interface Technology Laboratory, Virtual Environment Operating Shell (VEOS), provided much of the functionality of a distributed operating system in the way of programming language services [4]. However, we are primarily concerned with a single application and not a general purpose computing environment. Therefore, the most important questions about virtual environment software architectures we address here are:

- What is distributed?
- What are the modalities of the distribution?
- Why is it distributed?

1. Network Communication

Several aspects of network communication are largely responsible for answering the three questions above. The primary dimensions as shown for VEs are bandwidth, latency, distribution schemes, and reliability (Figure 3).

a. Bandwidth

We pay particular attention to the effect of bandwidth in this paper because the available network bandwidth determines the size and richness of a virtual environment. As the number of participants increases so do the bandwidth requirements. On local area networks (LANs), this has been not a major issue because technologies such as standard

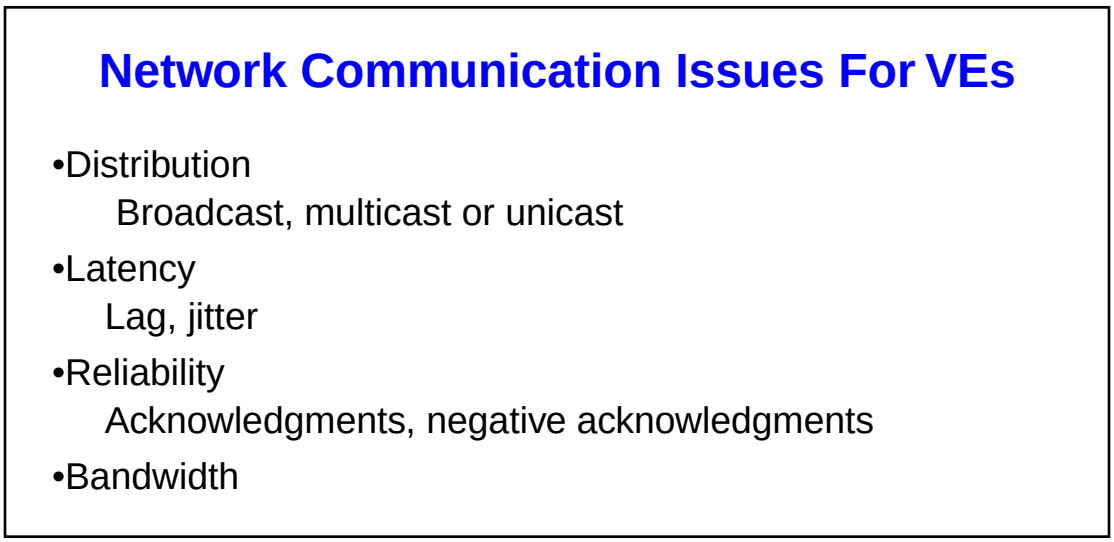

Figure 3. Communication issues.

Ethernet (10 Mbps) are relatively inexpensive and the number of users for LAN-based VEs has been limited. In contrast, for wide area networks (WANs) bandwidths have been generally limited to T1 (1.5 Mbps) but the potential user base is much larger through the Internet.

However, networks are now becoming fast enough to be true extensions to the computer's backplane and for the development of distributed VR applications. Distributed VR can require enormous bandwidth to support multiple users, video, audio and the exchange of 3D graphic primitives and models in real-time. Moreover, the mix of data requires new protocols and techniques to handle data appropriately over a network link. The technologies providing these gains in performance blur the traditional distinction between local area and wide area networks (LANs and WANs). There is also a convergence between networks that traditionally carried only voice and video over point-to-point links (circuit-switching) and those that handle packet-switched data.

The actual number of VEs to take advantage of these high speed networks have been small and have been associated with Grand Challenge (high performance computing) problems. The Multidimensional Applications and Gigabit Internetwork Consortium (MAGIC) network is a gigabit-per-second Asynchronous Transfer Mode (ATM) network that connects Minneapolis, Sioux Falls, Lawerence, Kansas, Kansas City and Ft. Leavenworth, Kansas. MAGIC is designed to allow a commander to see three-dimensional photo-realistic computer generated images of a very large area of interest in real time, both from ground level and from the air, using data stored in a remote database. These images will be generated from elevation data (Digital Elevation Maps), aerial photographs, models of buildings, and models of vehicles whose positions will be updated in real-time via the Global Positioning System. For example, a terrain database of Germany viewed in Kansas on a workstation receives images from California which are texture mapped onto the terrain in real-time [34]. The network provides trunk speeds of 2.4 Gbps and access speeds of 622

Mbps, allowing an application to use a supercomputer (CM-5) to process data from a database at a second location, and display the results on a workstation at a third location.

The NASA Computational Aerosciences Project is planning to use high speed networks to support visualization of large Computational Fluid Dynamics (CFD) data set by distributing processing onto several supercomputers across the United States. Gigabit networks will be required to move actual geometries generated by the supercomputers to be rendered on a remote graphics workstation [14].

Similarly, the Electronic Visualization Laboratory at the University of Illinois has used a combination of Ethernet, Fiber Distributed Data Interface (FDDI) and High-Performance Parallel Interface (HPPI) networks to develop a distributed VE application. The operator navigated through the VE using a CAVE (a system that projects images on three walls or hemi-cube for simulating "walkthroughs"), which was connected to an Silicon Graphics workstations used for rendering and control which in turn was connected to a CM-5 used for the actual simulation [26].

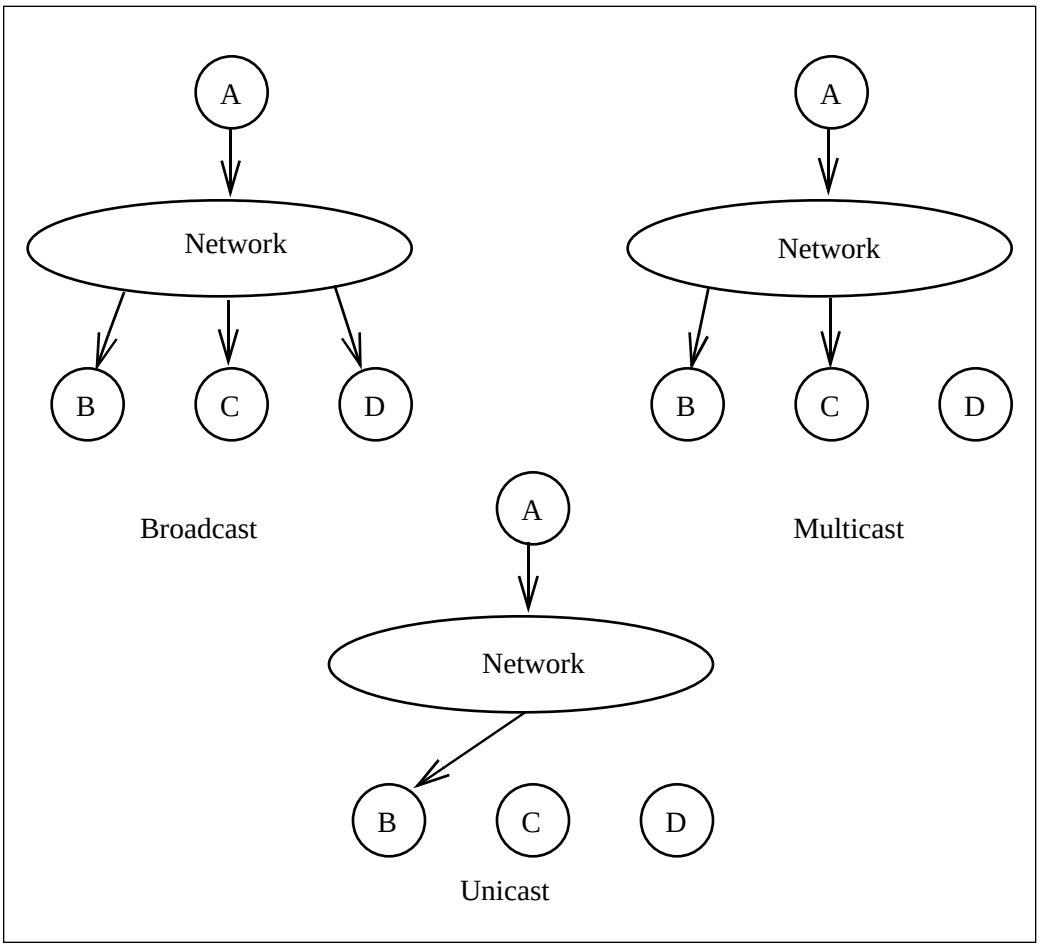

Figure 4. Examples of broadcast, multicast, and unicast.

b. Distribution

Some distribution schemes scale better than others. Three methods are shown in Figure 4. Multicast services allow arbitrarily-sized groups to communicate on a network via a single transmission by the source [19]. Multicast provides one-to-many and many-to-many delivery services for applications such as teleconferencing and distributed simulation in which there is a need to communicate with several other hosts simultaneously. For example, a multicast teleconference allows a host to send voice and video simultaneously to a set of (but not necessarily all) locations. With broadcast, data is sent to all hosts while unicast or point-to-point establishes communication between two hosts.

Most distributed VEs have employed some form of broadcast -- hardware-based or via the Internet Protocol (IP) -- or point-to-point communications. For example, the MR Toolkit Peer Package, which is used for creating distributed virtual reality applications over the Internet, uses unicast for communications among the applications though the developers have considered using IP Multicast [29]. Unicast is also the general approach for Grand Challenge applications like MAGIC. Another example is the NASA discussed above in which the network is a logical part of the visualization system much in a manner analogous to traditional image generators.

However, these schemes are bandwidth inefficient for large groups. Furthermore, broadcast, which is used in SIMNET -- an early military SIMulator NETwork developed by Bolt, Beranek, Newman (BBN) and Delta Graphics -- and most military distributed interactive simulation (DIS) implementations, are not suitable for internetworks because the network becomes flooded with unwanted traffic and it is difficult to avoid routing loops. Moreover, IP broadcast requires that all hosts examine a packet even if the information is not intended for that host, incurring a major performance penalty for that host because it must interrupt operations in order to perform this task at the operating system level. (SIMNET uses the hardware multicast capability of Ethernet but only to create a single multicast group for the entire distributed simulation.) Point-to-point requires the establishment of a connection or path from each node to every other node in the network for a total of N*(N-1) virtual connections in a group (see Figure 5). For example, with a 1000 member group, each of the 1000 individual hosts would have to separately address and send 999 identical packets.

Some researchers have proposed different ideas for using multicast to support virtual environments. The partitioning of virtual worlds into spaces is a common metaphor for VEs. Mitsubishi Electric Research Labs (MERL) have developed the Spline VE architecture. Spline uses multicast peer-to-peer communication. It also incorporates a region-based filtering scheme. These regions or “locales” partitioning the VE while “beacons” provide an entity updates about other objects in the VE [44].

Multi-User Dungeons (MUDs) have used the idea of spatialization. The *Jupiter* from Xerox PARC has extended the concept to associating “rooms” with multicast video and audio teleconferences [42]. Also at Xerox, Schilit and Theimer developed an active map service (AMS) that publishes the location of objects in a region using dynamic multicast groups associated with different parts of the region. For example, the system can track per-

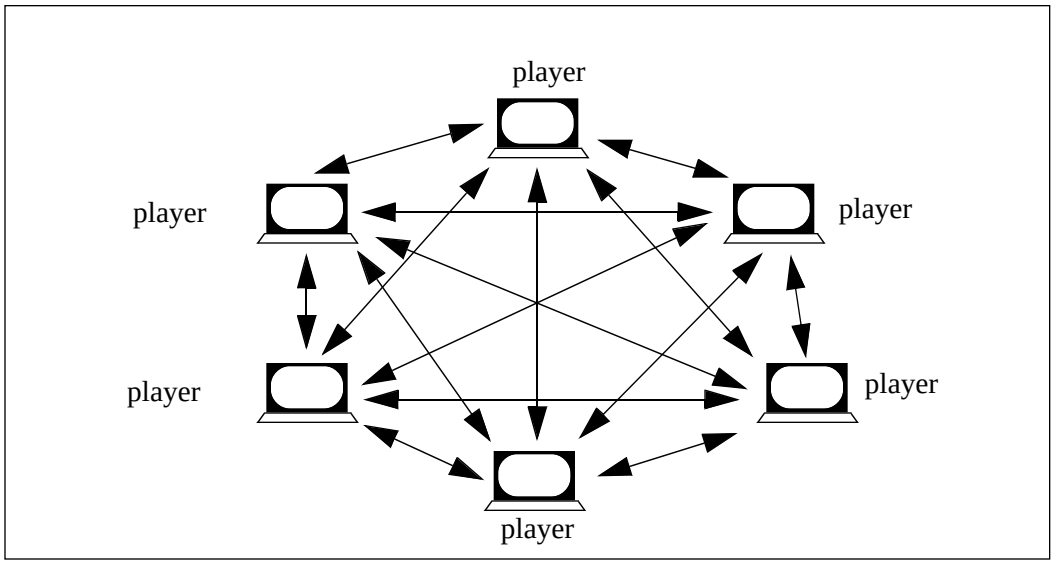

Figure 5. Distributed model.

sons in a building via the use of active badges. Using multicast for updates reduces aggregate message traffic [30].

Benford has described a concept for the spatial interaction of objects in a large-scale VE [2]. The spatial model uses different levels of awareness between objects based on their relative distance and mediated through a negotiation mechanism. The Swedish Institute of Computer Science's Distributed Interactive Virtual Environment (DIVE) -- an advanced experimental VE -- implements this concept using "standard VR collision detection" to determine when the transitions between awareness levels should occur [7].

Others have suggested using multicast for DIS but, very few have actually conducted research or implemented VEs using multicast communications. Stanford Research Institute recommended multicast in an early 1990 White Paper and it has been recommended for IEEE 1278 standards group [33].

Van Hook at the MIT Lincoln Laboratories has also proposed using a combination of grid-filtering to reduce the computational requirement of object filtering, an $O(n^2)$ operation [39]. Van Hook also suggested the idea of on-demand forwarding in which entities would send a low-rate broadcast with terse state information. Each receiver would compute a range check and send state data to the visible entities. However, object-filtering and on-demand forwarding essentially establish a multicast group for every receiver. For example, in Figure 6, Entity 1 and 2 join each other's multicast groups. Entity 3 is outside the range of 1 and 2 and therefore is a member of only its own group.

Until 1994, there was reluctance in the DIS community to use IP Multicast because of the Defense Advanced Research Projects Agency (DARPA) support for other network technologies, the lack of a software architecture and algorithms that could exploit it, and limited hardware support. The status of IP Multicast in the DIS world has changed. It is now part of the standard. Moreover, Mark Pullen at GMU and others have suggested using a two-level architecture using IP Multicast mapping to ATM multicast facilities [23].

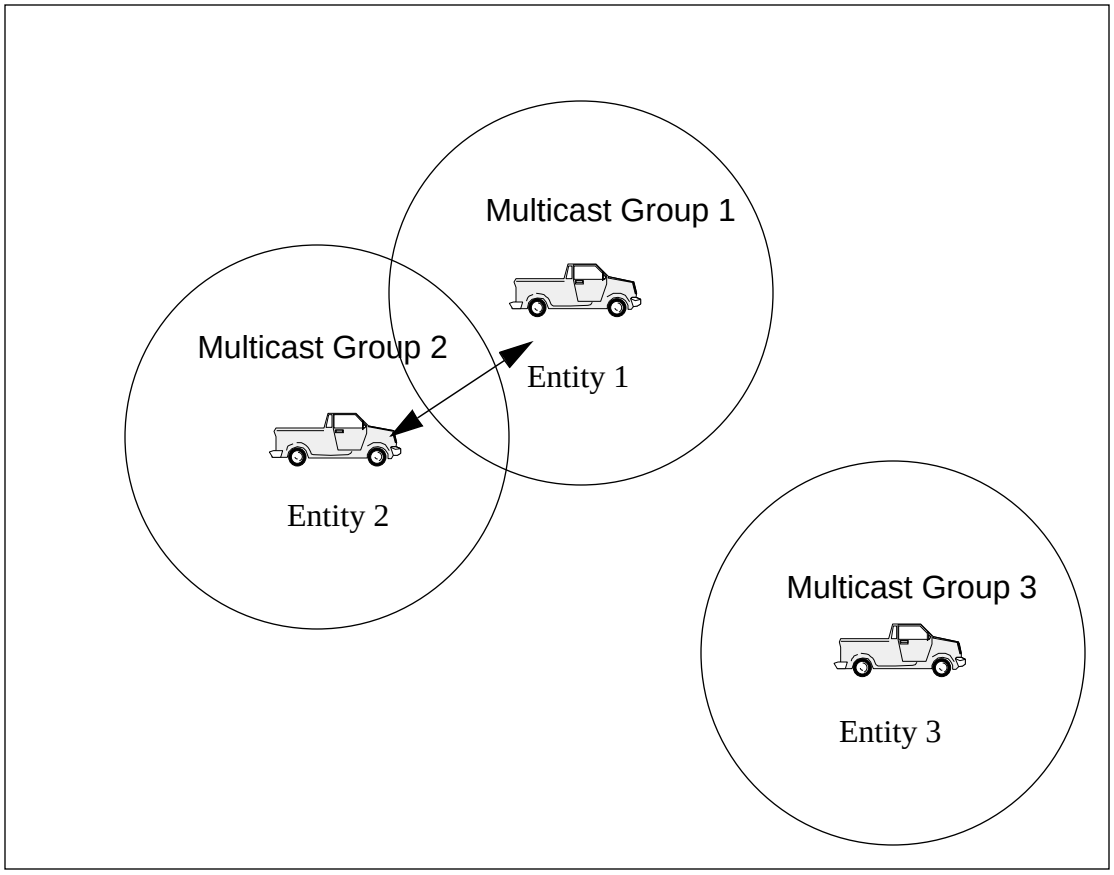

Figure 6. Object-based filtering.

c. Latency

Another dimension of communication is latency which controls the interactive and dynamic nature of the virtual environment -- how well the players mesh in behavior. If a distributed environment is to emulate the real world, it must operate in real-time in terms of human perception. A key challenge is that the appropriate systems involving human operators must deliver packets with minimal latency and generate textured 3D graphics at 30-60 Hz to guarantee the illusion of reality [41]. On top of this is the need to provide real-time audio, video, and imagery services for the simulation of player communication services.

Latency is a problem for network cue correlation. Sawler notes that both the delay of an individual cue (e.g., seeing an object move) and the variation in the length of the delay are important, particularly in closely coupled tasks which require a high degree of correlation (e.g., flying in formation) [27]. This becomes a major challenge in systems that use wide area networks because of delays induced by long paths, switches and routers. Network latency can be reduced to a certain extent by using dedicated links (or virtual ones

using protocols like the Reservation Protocol [43]), improvements in router and switching technologies, faster interfaces and computers.

However, more bandwidth is not necessarily a complete solution. Operating at gigabit speeds presents a new set of problems. New methods of handling congestion are required because of the high ratio of propagation time to cell transmission time [13]. By the time that a computer in New York sends a message telling a host in San Francisco to stop sending data, it is too late to have stopped a gigabit worth of information from being transmitted.

The bottlenecks will most likely be in the network interfaces, memory architectures and operating systems of the computers on either end. The slow progress in increasing the interface performance of FDDI is an example of the lag in technologies we will probably see as high speed networks are fully deployed. Nor have memory speeds kept up with the leaps made in CPU and network performance. At the operating system level, most VR applications are built on commercial versions of UNIX which are not designed for real-time performance.

Other methods are available for ameliorating the effects of latency. BBN developed dead-reckoning techniques for SIMNET which reduces communications loads on the network and perceived delays because of predictive modeling by the local host [15]. Briefly, a local entity passes state vectors to remote simulations. Both the local and remote simulations model the probable path of the entity. The local entity sends update state vectors when its current location or orientation exceeds some predetermined error threshold from the modeled path. Singhal has proposed a new dead-reckoning method that exploits position history [32].

However, lag can never be totally eliminated and for environments where the VE is widely distributed (e.g. Earth to Mars). Therefore, techniques such as synthetic fixtures are used which provide force and visual clues to operators in limited domains about that environment [28].

d. Reliability

Finally, communications reliability often forces a compromise between bandwidth and latency. Reliability means that systems can logically assume that data sent is always received correctly, thus obviating the need to periodically re-send the information. Unfortunately, to guarantee delivery, the underlying network architecture must use acknowledgment and error recovery schemes that can introduce large amounts of delay - a common case on WANs and with large distributed systems. Additionally, some transport protocols such as the Transport Control Protocol (TCP) use congestion control mechanisms that are unsuitable for real-time traffic because they throttle back the packet rate if congestion is detected.

Reliable multicast protocols are currently not practical for large groups because in order to guarantee that a packet is properly received at every host in the group, an acknowledgment and retransmission scheme is required [18]. With a large distributed simulation, reliability, e.g., as provided in TCP, would penalize real-time performance merely by having to maintain timers for each host’s acknowledgment and by holding up

flow when a packet is lost for retransmission. Flow control introduces delay to the network to reduce congestion. Therefore, it is also not appropriate for DIS which can recover from a lost packet more gracefully than from late arrivals -- it is impossible for real-time simulations to go backward in time. For example, when a packet is lost the receiving host notifies the sender, possibly invalidating a number of packets already sent because of propagation and network processing delay. The sender must retrieve a copy of the packet lost and retransmit it. This also affects the windowing behavior which in turn slows throughput.

However, researchers are trying to develop both a reliable and *scalable* multicast service. The ISIS system, developed by Ken Birman, uses a reliable multicast service to guarantee that the virtual environment databases are accurately and synchronously replicated [7]. (A recent version of ISIS implements a reliable transport layer on top of IP Multicast). However, a peer group with more than twenty or thirty members is about as large as can be efficiently supported by ISIS.

Brian Whetten and Simon Kaplan developed the Reliable Multicast Protocol (RMP) which is based on a token ring protocol that sits atop IP Multicast. This method uses sequencing and negative acknowledgments (NACKs)[40]. The problem with this method is the potential for NACK implosions, in which a group of receivers simultaneous send NACKs, adding to congestion and consequently causing the loss of more packets which introduce more NACKs. Again, reliable systems are not likely to operate in real-time. As Partridge in [18] states, “the problem of reliable multicasting over internets has not been solved”.

Netrek, an Internet multiplayer game that uses X Window system graphics, took the approach of using different degrees of reliability to gain better real time performance [16]. (Mark Pullen of GMU has suggested a similar concept for DIS called the Selective Reliability Transport Protocol [22].) Previous versions of the game used TCP. New versions have a protocol that:

- guarantees the reliability of certain packets with TCP such as error conditions and session setup and for information that is sent infrequently (server message of the day)
- does not guarantee reliability for frequent and noncritical data such as player state (speed, direction)
- allows switching on demand from TCP to UDP/TCP and back
- won’t hang or cause abnormal termination if a UDP packet is lost

2. Views

Views are the windows into the virtual environment from the perspective of the people or processes who use it. We define two kinds of useful views for distributed environments. The first one is the synchronous view. An example of this is in a distributed flight simulator

where one machine controls the forward image, and two other hosts each process the left and right cockpit window perspectives. The images are coordinated to give the illusion that they are all part of single cockpit view. Synchronism requires both high reliability and low latency. Therefore, virtual environments that require synchronous views are for practical reasons restricted to local area networks. An example of such an environment is the RAVEN simulator developed by Southwest Research Institute for NASA synchronizes shuttle astronaut viewpoints which are rendered on different machines to improve rendering performance [8]. The CAVE uses a similar approach to synchronize each image frame projected on the screen of a hemi-cube with the added component of synchronizing the simulation run separately on a CM-5. Originally, this was done using a SCRAMNET (proprietary fiber optic, shared memory LAN). Later, this was accomplished using multiple raster managers on an SGI Reality Engine Onyx and shared memory.

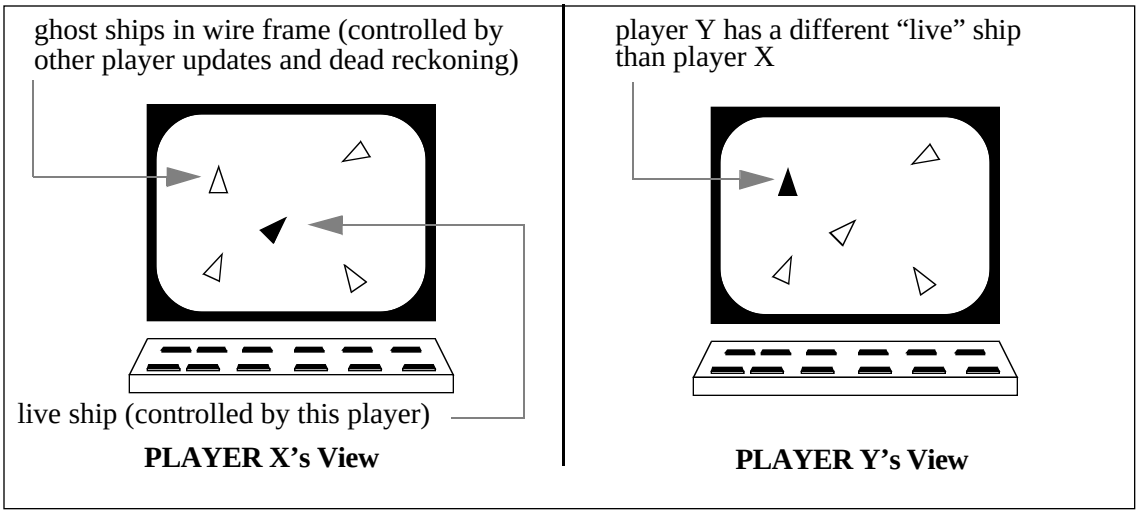

Figure 7. Two views of the simulation.

Synchronous views are also important for computer-aided design and systems used for concurrent engineering. The Fraunhofer Institute for Computer Graphics has developed a virtual prototyping environment that implements a shared-3D CAD viewer [45]. The VE is designed to demonstrate the ability of engineering teams to work globally via the Fraunhofer Transatlantic Research and Development Environment (TRADE).

The second and most general concept is the asynchronous view. In this paradigm, multiple users have individual control over when and what they can see in the virtual environment concurrently (Figure 7). Participants can be physically separated over a local area network or a wide area network. Their awareness of each other's presence, if they are represented by an object, is brought about *inside* the virtual environment. The Naval Postgraduate Schools VE, NPSNET, supports the DIS protocol and uses the asynchronous model where each view is typically that of the simulated entity [46]. Views not associated with an entity are often referred to as "magic carpets" or "stealth" vehicles. Stealth entities only "listen" to the distributed world traffic because there is no need for the world to have knowledge of the viewer. Large-scale VEs will use asynchronous views because of the cost of synchronization over wide-area networks. Synchronous views will be important for

small VEs in which precise cooperative manipulation of objects is required and for applications or device communication distributed over LANs.

3. Data

Perhaps the most difficult decision of building a distributed environment is determining where to put the data relevant to the state of the virtual world and its objects. These decisions affect the scale, communication requirements, and reliability of the VE data. For example, a real-time system requiring strong consistency will be inherently difficult to scale because of the need for causality and automaticity. For now, at least, large VEs only allow weak consistency among group members.

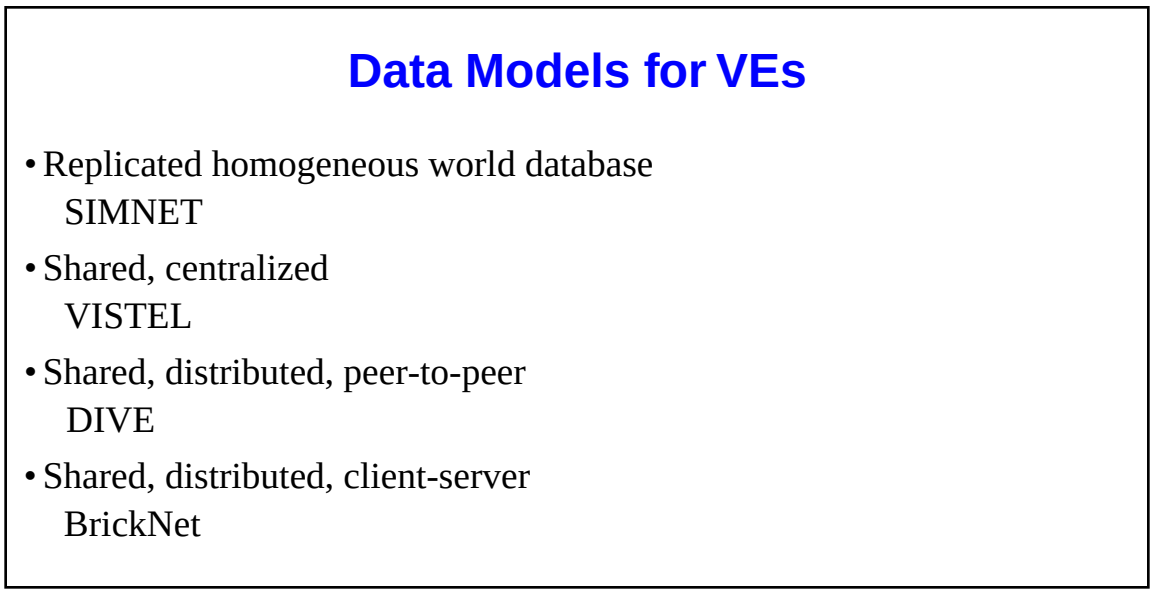

Figure 8. Data models for VEs.

There are many conceivable ways of distributing persistent or semi-persistent data (see Figure 8). We present some of the most prevalent methods in current VEs:

a. Replicated homogeneous world

A common method for large VEs is to initialize the state of every system participating in the distributed environment with a homogeneous world database containing information about the terrain, model geometry, textures, and behavior of all that is represented in the virtual environment. Communicated among all the users of the environment are object state changes such as vehicle location or events such as the detonation of a simulated missile or collisions between two objects. The advantage of this approach is that messages are relatively small. The disadvantages are that it is relatively inflexible and that as virtual environment content increases so must everyone's database. Moreover, over time, the world becomes inconsistent among the participants through the loss of state and event messages. This is the model for SIMNET. However, once a

simulation begins, each host maintains its own database without making any effort at guaranteeing consistency except through the use of "heartbeat" messages and event updates [21].

b. Shared, centralized databases

On the other hand, the Virtual Space Teleconferencing System (VISTEL) uses a shared world database. As its name implies, VISTEL is a teleconferencing system that displays 3D models of each conference participant. Changes in a model's shape, reflecting changes in a person's facial expression, are sent via messages to a central server and redistributed. Only one user at a time can modify the database (see Figure 9)[17].

This is the model used by MUDs except that they typical employ relatively primitive clients. Text-based MUDs use TCP connections to a central server that does almost all the computation and maintains the state of the VW. For text communication, this typically scales to about fifty concurrent users who "move" about among rooms, create and delete new objects or actions, and communicate with each other. LamdaMOO from Xerox Palo Alto Research Center is probably the most advanced MUD [42]. Using a centralized server for 3D virtual worlds is obviously limited to a few participants because of input/ output (I/O) contention, and maintaining a dynamic object database in real-time.

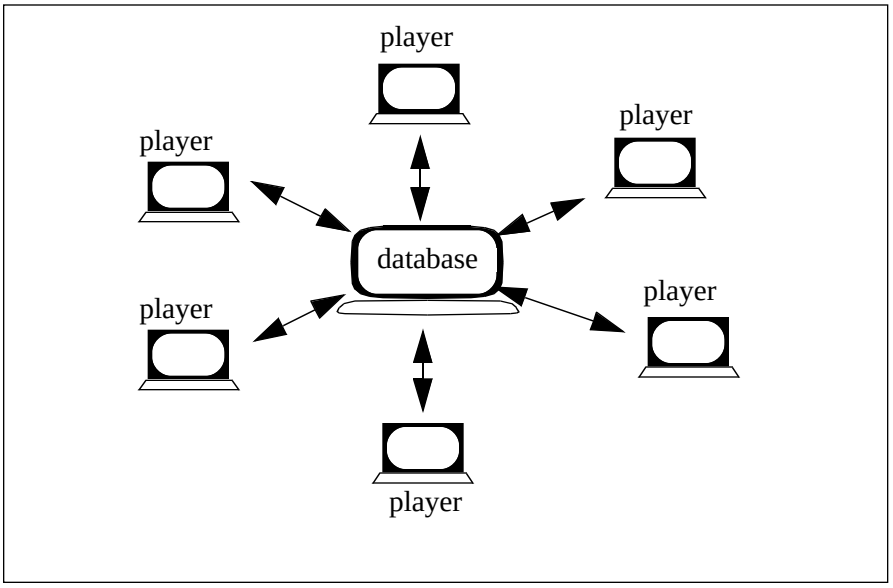

Figure 9. Centralized model.

The I/O problem is demonstrated in Netrek which scales to about 18 players with UDP and uses an asymmetric communications model [16]. The data in Figure 10 from J. Mark Noworolski shows how the server becomes the bottleneck because it must retransmit all other players's state to each client. In this case communication from individual clients are only 168 bytes per second. The server, on the other hand, must take every client message and redistribute to it to all the other clients with an order of magnitude increase in bandwidth required.

Server -> Client network usage:
Maximum CPS during normal play: 3588 bytes per second
Standard deviation: 918
Total bytes received 1795888, average CPS: 803.0

Client -> Server network usage:
Maximum CPS out during normal play: 168 bytes per second
Standard deviation out: 21
Total bytes sent 20580, average CPS: 18.0

Figure 10. Server vs. client communication in Netrek.

c. Shared, distributed databases with peer-to-peer updates

Many distributed systems strive to simulate shared memory architectures. For example, DIVE has a homogeneous fully-replicated distributed database. However, unlike SIMNET the entire database is dynamic and uses reliable multicast protocols to actively replicate new objects. A disadvantage with this approach is that it is difficult to scale up because of the communications costs associated with maintaining reliability and consistent data across wide area networks. Modeling complex or dense objects, such as constructing a large CAD model or changing a terrain database, is very expensive (though highly desirable) in terms of the number of polygons that might be created, changed, and communicated over a network.

Virtual environments that use Linda, the parallel programming language, also trade performance for a relatively simple blackboard programming model. For example, Denis Amselem of SRI developed a VE using Linda with an unusual hand-held interface -- a portable LCD television with a space tracker for VE navigation. Performance of the multiuser system limited it to three participants [1]. The simplicity and illusion of shared memory presented by Linda is also the reason why this system suffered from poor performance. Data must reside *somewhere.* In this case, it was on a central server.

d. Shared, distributed, client-server databases

Another technique is to use a variant of the client-server model in which the database is partitioned among clients and communication is mediated by a central server. For example, in BrickNet, as an entity moves through the virtual environment, its database is updated by an object-request broker on a server that has knowledge of which client maintains that part of the world [31]. BrickNet may be most appropriate for large CAD environments because it attempts to tackle the walkthrough problem of a virtual environment that has huge numbers of component models and provides multiple views simultaneously to a group of users. However, in a dynamic large scale world, the servers

can quickly become I/O bottlenecks, increasing the inherent latency of the virtual environment.

In a similar approach to BrickNet and DIVE, the Model, Architecture and System for Spatial Interaction in Virtual Environments (MASSIVE) system uses a spatial model for data partitioning among clients. In this case, an entity declares its world to a local "aura" manager which in turn informs other aura collision managers. These managers broker between objects by detecting proximal collisions and informing each of the peer entities of mutual interface references[12].

Pure client-server systems that strictly use classic remote procedure calls (RPC) do not scale well for a number of reasons. Partridge points out that the RPC is poorly suited for high speed networks because communication is achieved by sending a message and waiting for a reply. As relative delay of networks increases, RPCs become expensive [18].

4. Processes

Distributing processes to multiple hosts increases the aggregate computing power associated with a simulation. We can use this not only to provide the capability to distribute views but also handle a variety of input devices. SIMNET and its descendants, such as DIS-compliant systems, also make use of the aggregate computing power by taking advantage of dead-reckoning to reduce the need for network communication.

With the exception of the DIS model, practically all distributed environments assume that the same kind of processes are running on each host that has the same function (architectures may differ). The advantage of this approach is consistency. The disadvantage is that it is very inflexible. DIS is a protocol designed so that different developers can create different simulations on different machines that theoretically can share in the same virtual environment because they can communicate at some common level. The problem with this is that no protocol is complete and DIS is not an exception. For example, new objects cannot be introduced without a change in the standard.

The AVIARY system has homogenous processes but contains Object Servers which permits migration of lightweight objects to enable load balancing. These objects represent the entities and the processes which control them. DIVE uses the concept of process groups from ISIS to partition the VE into rooms or spatial regions. The MR Toolkit distributes processes that support different components of the VE such as the input devices [29]. It provides an interpreted language, the Object Modeling Language (OML) that allows platform independence for developing virtual environments. OML specifies the behavior and geometry of VE entities. Similarly, BrickNet uses a language called Starship. BrickNet objects can share or transfer behaviors that are specified in Starship. These behaviors are either environmentally-dependent, reactive, or capability based.

Other more general-purpose scripting languages may provide the capability to migrate processes and objects across diverse platforms by using active messaging. Sun's Java is the primary example of this class. A byte-compiled, interpretive language, similar to C++, Java is meld to the client-server architecture of the World Wide Web. Java also supports peer IP Multicast communication and in the future a host of multimedia components.

Gavin Bell and Tony Parisi of SGI have developed the Virtual Reality Modeling Language (VRML) which "is a language for describing multi-participant interactive simulations -- virtual worlds networked via the global Internet and hyperlinked with the World Wide Web" [20]. However, VRML merely describes a 3D scene and methods for interacting with models. Though VRML 2.0 allows the use of Java to provide object behaviors, VRML itself does not provide a mechanism for communication among distributed users.

The proprietary Telescript language is an innovative communication technology that -- like Java -- also considers the network as a platform, on top of which you can run applications that are not bound to a specific node of the network, but, to the contrary, are intended to move around the network during their execution. Telescript itself is an interpreted language that is specifically designed for communication. It provides primitives that allow the script to suspend, migrate to another node of the network, and resume execution from the same point. The key idea is procedural messaging. With Telescript, write agents are sent around the network to accomplish the tasks you want. Instead of having a client dealing with a server by means of a set of messages sent back and forth, you build an agent and send it where the server is. The agent is smart enough to interact with the server, and returns to the sender with the required information. In this way, bandwidth consumption is reduced and we can build agents that seek information on our behalf [24].

Safe-Tcl is a language for active or agent-based mail in which the data delivered through the mail constitute a program in a well- specified language, allowing the program to be automatically evaluated on behalf of the recipient when the mail is "read." The syntax of Safe-Tcl is identical to the syntax of Tcl [3]. No syntactic constructs are changed. The only difference, therefore, between Tcl and Safe-Tcl is the set of available primitive functions and procedures. Safe-Tcl may be described as an "extended subset" of Tcl, in that the "dangerous" primitives in Tcl have been removed, while certain new primitives have been added. An example of a dangerous primitive is an exec call in Tcl which can start up a new process. Eliminating this from Safe-Tcl helps avoid the possibility that a script could generate unwanted behavior by the receiving host. DIVE uses Tcl.

In a distributed VE, clients can be homogeneous as is the case for DIVE. Clients can also be dissimilar except for the communications protocols among them, providing interoperability (e.g., DIS and SIMNET systems, which exchange standard state and event data). Furthermore, processes can be designed to migrate across homogeneous architectures like AVIARY. New scripting languages like Safe-Tcl offer the opportunity for migrating processes across heterogeneous systems, therefore permitting efficient exchange of object behaviors as well as entity state within large, heterogenous VEs.

SUMMARY

We have discussed VEs in the context of how communications, views, data, and processes are distributed. We have not exhausted all the considerations for developing VEs but have emphasized those aspects critical to scaling environments. Most of the systems described here scale to accommodate a handful of users.

We also know that systems that demand strong data consistency, causality, and reliable communications and at the same time need to support real-time interaction are not like-

ly to scale well. If the systems are to be geographically dispersed, then high-speed, multicast communication is required.

Replicated world databases are more communication efficient than centralized or distributed shared database schemes. However, they generally lack a way for maintaining world consistency -- a problem with unreliable transport mechanisms like UDP. They also lack the ability to update the VE with new objects or behaviors. However, large VEs could use a mixed model -- client initialization with small replicated data sets and a distributed client-server model. This would allow more data consistency and persistence if a mechanism or heuristic is used to reduce transfer latency.

We offer the conjecture that the current trend toward a pure client-server paradigm of the World Wide Web in VEs is a future dead-end unless hybrid architectures that incorporate peer communications are used. New languages like Java and VRML will provide innovative methods for building virtual worlds. However, the problem of achieving scalability, reliability, and real-time interaction simultaneously will not likely be resolved soon.

ACKNOWLEDGMENTS

A grateful thanks Rich Gossweiller for the use of his figures. This work would not have been possible without the support of our research sponsors: DARPA, DMSO, USA STRICOM, USA TRAC.

The paper was originally presented at the The Second IEEE Workshop on Networked Realities, Boston, Massachusetts, 26-28 October, 1995.

RESOURCES

The NPSNET Research Group's WWW home page *http://www.cs.nps.navy.mil/research/npsnet.* The CRCG VR Center home page is at *http://www.crcg.edu.*

1. Amselem, Denis."A Window on Shared Virtual Environments," Presence, 4, 2, Spring 95.
2. Benford, S., Bowers, J., Fahlen, L. and Greenhalgh, C., "Managing Mutual Awareness in Collaborative Virtual Environments", In Proceedings of VRST94, World Scientific Publishing Company, NJ, pp. 223-236.
3. Borenstein, Nathaniel and Rose, Marshall T.," MIME Extensions for Mail-Enabled Applications," Internet working draft, ftp://thumper.bellcore.com/pub/nsb/st/safe-tcl.txt, October 1993s.
4. Bricken, William and Coco, Geoffrey, "The Veos Project", ftp.u.washington.edu:/public/VirtualReality/HITL/papers/tech-reports/Veos_Project.ps, 1993.
5. Burka, Lauren, "The MUD Archive, "http://www.ccs.neu.edu/home/lpb/muddex.html.
6. CACI Products Company," MODSIM II Reference Manual," La Jolla, CA, 1991.
7. Carlsson, C. and Hagsand, O. 1993, "DIVE - a Multi User Virtual Reality System," Proceedings of VRAIS93, September 18-22, Seattle, WA IEEE, NJ, 1993. pp. 394-400.
8. Cater, John P, "Use of the Remote Access Virtual Environment RAVEN for Coordinated IVA-EVA Astronaut Training and Evaluation," Presence, 4,1, Winter 1994.
9. Durlach, Nathanliel I. and Mavor, Anne S., Virtual Reality: Scientific and Technological Challenges, National Academy Press, Washington, D.C. 1995.
10. Gisi, Mark A., Sacchi, Cristiano, "Co-CAD: A Multi-user Collaborative Mechanical CAD System," Presence, 3, 4, Winter 1994.
11. Gossweiler, Rich, Laferriere Robert J., Keller, Michael L. and Pauch, Randy, "An Introductory Tutorial for Developing Multiuser Virtual Environments," Presence, 3, 4 Winter 1994.

12. Greenhalgh, Chris, and Benford, Steve, “MASSIVE: a Collaborative Virtual Environment for Tele-Conferencing,” submitted to ACM TOCHI, 1994.
13. Habib, Ibrahim W. and Saadawi, Tarek N., “Controlling Flow and Avoiding Congestion in Broadband Networks,” IEEE Communications Magazine, Vol. 29, No. 10, October 1991, pp. 46-53.
14. McCabe, James D., “Data Communications Required for CAS Applications,” Presence, 4, 2 Summer 1995.
15. Miller, Duncan C.,Pope, Arthur C. and Waters, Roland M, “Long-Haul Networking of Simulators,” Proceedings: Tenth Interservice/Industry Training Systems Conference, Orlando, Florida, December 1989, p. 2.
16. Netrek, http://www.cs.cmu.edu:8001/afs/cs/user/jch/netrek/udp.
17. Ohya, Jun, Kitamura, Yasuichi, Takemura, Haruo, et. al., “Real-time Reproduction of 3D Human Images in Virtual Space Teleconferencing,” Proceedings of IEEE Virtual Reality International Symposium, September 1993, pp. 408-414.
18. Partridge, Craig, Gigabit Networking, Addison-Wesley, Reading, Massachusetts, 1994.
19. Perlman, Radia, Interconnections: Bridges and Routers, Addison-Wesley, New York, 1992, p. 258.
20. Pesce, M., “The Virtual Reality Modeling Language,” http://www.eit.com/vrml/vrmlspec.html.
21. Pope, Arthur, “The SIMNET Network and Protocols,” BBN Report No. 7102, BBN Systems and Technologies, Cambridge, Massachusetts, July, 1989.
22. Pullen, Mark, “Toward a Requirement Specification for A Selectively Reliable Transport Protocol,” DIS Communications Architecture Subgroup Winter Workshop, 18 January 1995.
23. Pullen, Mark, “Dual-Mode Multicast,” DIS Communications Architecture Subgroup Winter Workshop, 18 January 1995.
24. Reinhardt, Andy, “The Network With Smarts,” BYTE, October 1994.
25. Rich, C. et al., “Demonstration of an Interactive Multimedia Environment,” IEEE Computer, Vol. 27, No. 12, Dec. 1994, pp. 15-22
26. Roy, Tina, M., Cruz-Niera, Carolina, DeFanti, Thoma A., Sandin, Daniel J., “Steering a High Performance Computing Application from a Virtual Environment,” Presence, 4,2 Summer 1995.
27. Sawler, Robert, Matusof, 'Issues Concerning Cue Correlation and Synchronous Networked Simulators,' AIAA, 1991.
28. Sayers, Craig, and Paul, Richard, 'An Operator Interface for Teleprogramming Employing Synthetic Fixtures,' Presence, 3, 4, Winter 1994, pp. 309-320.
29. Shaw, Chris, and Green Mark, 'The MR Toolkit Peers Package and Experiment,' Proceedings of IEEE Virtual Reality International Symposium, September1993, pp. 463-469.
30. Schilit, Bill N. and Theimer, Marvin M, “Disseminating Active Map Information to Mobile Hosts,” IEEE Network, September 1994, pp. 22-32.
31. Singh, Gurminder, “A Software Toolkit for Network-Based Virtual Environments,” Presence, 3, 1, Summer1994.
32. Singhal, Sandeep K., “Using a Position History-Based Protocol for Distributed Object Visualization.,” in Designing Real-Time Graphics for Entertainment [Course Notes for SIGGRAPH ‘94 Course #14] July 1994.
33. Stanford Research Institute International, ATD-1 Architecture White Paper Edit Draft, Menlo Park, CA, undated.
34. Stanford Research Institute MAGIC Home Page, http://www.ai.sri.com:80/~magic/.
35. Taubes, Gary, “Surgery in Cyberspace,” Discover, December 1994, pp. 85-94.
36. “The Isis Distributed Toolkit Version 3.0 User Reference Manual,” Isis Distributed Systems. 1992, pp. 3-9.
37. defaultstricom.html.
38. Valdes, Ray, “Introducing ScriptX,” Dr. Dobb’s Journal, Vol. 19, Issue 13, November 1994.
39. Van Hook, Daniel J., Calvin, James O, “Approaches to Relevance Filtering,” 11th DIS Workshop, September 1994, pp. 367-369.
40. Whetten, Brian and Kaplan, Simon., “A High Performance Totally Ordered Multicast Protocol,” Submitted to SIGCOMM’94.
41. Wloka, Mathias M., “Lag in Multiprocessor VR,” Presence, 4, 1, Spring 1995.
42. Curtis, P., Nichols, D.A, “MUDs Grow Up: Social Virtual Reality in the Real World,” 1994, ftp://ftp.parc.xerox.com/pub/MOO/papers/MUDsGrowUp.ps.
43. Deering, Stephen, “MBONE-The Multicast Backbone,” CERFnet Seminar, 3 March 1993.
44. Barrus, J.W., Waters, R.C., and Anderson D.B., Locales and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments”, IEEE Virtual Reality Annual International Symposium, Sant Clara, CA, March 1996, 204--213.
45. Anderson, Brian G., Uwe Jasnoch, Joseph, Hans, “Coconut - A Virtual Protoyping Environment”, Computer Graphik Topics, March 1996, Vol. 8, p. 20-22.

46. Macedonia, Michael R., Zyda, Michael J. ,, Pratt, David R., Barham, Paul T., Zeswitz, Steven. “NPSNET: A Network Software Architecture for Large Virtual Environments”. Presen. Vol 3, No. 4.